Distributed Bayes Blocks for Variational Bayesian Learning

Antti Honkela
Adaptive Informatics Research Centre, Helsinki University of Technology, Finland

1 Introduction

Recent trend in development of computer hardware points toward increased parallelism instead of speeding up individual processor cores. This introduces new challenges to developers of computationally intensive software. This includes Bayesian inference and learning that often require extensive computation for anything but the simplest models.

In this work preliminary results on a distributed version of Bayes Blocks software library [1, 4] are presented. Bayes Blocks is a software implementation of the variational Bayesian building block framework [7] that allows automated derivation of variational learning procedures for a variety of models, including nonlinear and variance models. The library is implemented in C++ with Python bindings. The underlying framework resembles the variational message passing (VMP) framework [8], but includes support for nonlinear and other models that are not in the conjugate exponential family.

Our aim is to develop a method that would allow learning the model in a distributed, asynchronous fashion. Asynchronous mode of operation promises greatest benefits from parallelisation but makes guaranteeing stability of the algorithm challenging [3].

2 Variational Bayesian Learning and Bayes Blocks

Variational Bayesian learning is based on approximating the posterior distribution of the latent variables $\boldsymbol{S}$ and model parameters $\boldsymbol{\theta}$ with a simpler distribution. Bayes Blocks uses a mean field type factorial approximation $q(\boldsymbol{S}, \boldsymbol{\theta}) = q(\boldsymbol{S})q(\boldsymbol{\theta})$ [2], that is fitted by maximising the lower bound

$$\mathcal{B} = \left\langle \log \frac{p(\boldsymbol{S}, \boldsymbol{\theta}, \boldsymbol{X}|\mathcal{H})}{q(\boldsymbol{S}, \boldsymbol{\theta})} \right\rangle = \log p(\boldsymbol{X}|\mathcal{H}) - D_{KL}(q(\boldsymbol{S}, \boldsymbol{\theta})||p(\boldsymbol{S}, \boldsymbol{\theta}|\boldsymbol{X}, \mathcal{H})) \leq \log p(\boldsymbol{X}|\mathcal{H}) \tag{1}$$

of the marginal log-likelihood $\log p(\boldsymbol{X}|\mathcal{H})$. Here $\langle\cdot\rangle$ denotes expectation over $q(\boldsymbol{S}, \boldsymbol{\theta})$ and $D_{KL}(q||p)$ is the Kullback–Leibler divergence between q and p.

Bayes Blocks models are defined by connecting variable nodes that represent latent and observed variables and computational nodes for mathematical operations such as sum, product and nonlinearities. For continuous models the variables are mainly Gaussian with log-normal variance parametrisation. These blocks can be used to build a wide variety of linear and nonlinear models such as hierarchical nonlinear factor analysis [6] and hierarchical variance models or heteroscedastic models [5].

The (variational) parameters of $q(\boldsymbol{S}, \boldsymbol{\theta})$ are updated using a variational EM algorithm that uses the gradients of $\mathcal{B}$ with respect to the parameters. These can be evaluated efficiently using the network structure and a message passing algorithm. As the form of the potential causing the gradients is known, closed form update rules can be derived for most of the variables.

3 Distributed Bayes Blocks

A Bayes Blocks model can be distributed to several computers by adding new computational nodes that encapsulate the network connection. These nodes operate by default asynchronously by reporting the previously received values and gradients. In order to stabilise the method, the gradients are transformed to be invariant of the value of parent with respect to whom they are evaluated. This prevents placing product nodes as a descendant of a network connection before a variable node. Otherwise the network connections can be placed freely. In the current implementation, the actual network connections are handled using TCP/IP, but more sophisticated transports could be easily incorporated as well.

Direct implementation of this method would still be unstable for many models. If, for instance, the sum of two variables residing on different computers is used to model something, the asynchronous updates will make both variables correct the same error in the output of the sum and overshoot, leading to divergent updates. This can be corrected by adding a momentum term to the reported gradients so that the internal gradient value g_t held by the network connection node when receiving a new value g_{new} over the network is updated using the old value g_{t-1} as $g_t = (1-\alpha)g_{\text{new}} + \alpha g_{t-1}$ for some $\alpha \in [0,1]$. Even relatively small values such as $\alpha = 0.1$ are sufficient to effectively stabilise the iteration in the above example.

4 Experiment

The distributed Bayes Blocks library was tested with a distributed factor analysis model

$$\mathbf{x} = \mathbf{A}\mathbf{s} + \boldsymbol{\epsilon}, \tag{2}$$

where $\mathbf{x}$ denotes the observations, $\mathbf{A}$ is the loading matrix, $\mathbf{s} \sim N(\mathbf{0}, \mathbf{P})$ are the factors and $\boldsymbol{\epsilon} \sim N(\mathbf{0}, \mathbf{Q})$ the noise. The covariance matrices $\mathbf{P}$ and $\mathbf{Q}$ were assumed to be diagonal. 100 samples of 4 dimensional data were drawn from the distribution specified by the model with two factors and $\mathbf{P} = \mathbf{I}, \mathbf{Q} = .01\mathbf{I}$, and $\mathbf{A}$ drawn randomly from $N(0, 2\mathbf{I})$.

The model used for the data was similar to the one specified above but with 3 factors. The prior for $\mathbf{A}$ was fixed to unit variance while the diagonal terms of the covariances of $\mathbf{s}$ and $\boldsymbol{\epsilon}$ were estimated from the data. The model was split to four independent processes communicating over the network so that each factor and its variance as well as the corresponding column of the mixing matrix were in separate processes and the data and noise covariance in the last. The structure of the model and its distribution to different processes is illustrated in the left panel of Fig. 1.

The convergence of the estimates of selected parameters is shown in the right panel of Fig. 1. In order to break the symmetry between different factors and avoid problems caused by unsynchronised starting times of different processes, the factors have random initialisations that decay during the first 500 iterations. As the figures show, the model converges rather quickly after the effect of the initialisations has been removed. The final slow drift is caused by rescaling of $\mathbf{A}$ and $\mathbf{s}$ to make $\mathbf{A}$ confirm to its fixed prior. The directions of the columns of $\mathbf{A}$ do not change after the first 1000 iterations. The extra third factor has been essentially shut down by making the corresponding column of $\mathbf{A}$ equal to zero and the variance of $\mathbf{s}$ small.

5 Conclusion

The presented distributed Bayes Blocks method can successfully divide the computation required for variational Bayesian learning of large models. The method was demonstrated with a proof of concept distributed factor analysis model. The ability to divide a model to several computers can be helpful in many cases, such as when the model is too large to fit to memory of a single computer.

While the method works in the simple example, much work is needed before it is really practical. The variational EM employed often converges slowly and regular Bayes Blocks implements certain speedups.

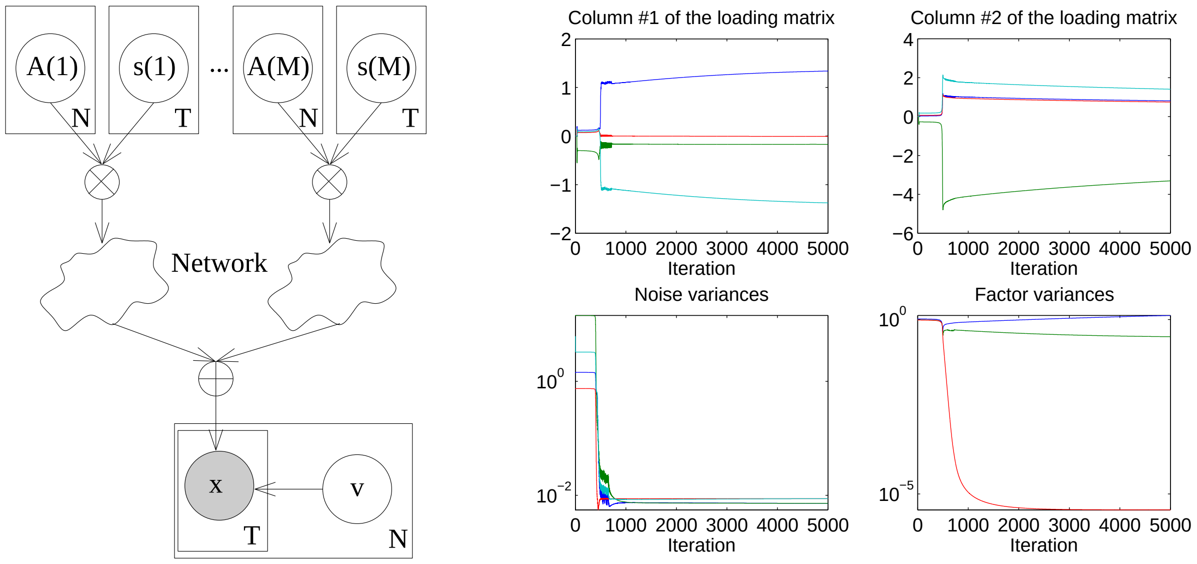

Figure 1: Left: An illustration of the distributed factor analysis graphical model. The variances of the factors have been omitted from the figure. Right: Convergence of selected parameter estimates. The values show the means of the parameters in the posterior approximation q.

The applicability of these to distributed models is an important open issue. Further research is also needed to provide guarantees for the stability of the method.

References

[1] M. Harva, T. Raiko, A. Honkela, H. Valpola, and J. Karhunen. Bayes Blocks: An implementation of the variational Bayesian building blocks framework. In *Proc. 21st Conf. on Uncertainty in Artificial Intelligence (UAI 2005)*, pages 259–266, Edinburgh, Scotland, July 2005.

[2] T. S. Jaakkola. Tutorial on variational approximation methods. In M. Opper and D. Saad, editors, *Advanced Mean Field Methods: Theory and Practice*, pages 129–159. The MIT Press, 2001.

[3] A. Pfeffer and T. Tai. Asynchronous dynamic Bayesian networks. In *Proc. 21st Conf. on Uncertainty in Artificial Intelligence (UAI 2005)*, pages 467–476, Edinburgh, Scotland, July 2005.

[4] H. Valpola, A. Honkela, M. Harva, A. Ilin, T. Raiko, and T. Östman. Bayes blocks software library. *http://www.cis.hut.fi/projects/bayes/software/*, 2003.

[5] H. Valpola, M. Harva, and J. Karhunen. Hierarchical models of variance sources. *Signal Processing*, 84(2):267–282, 2004.

[6] H. Valpola, T. Östman, and J. Karhunen. Nonlinear independent factor analysis by hierarchical models. In *Proc. 4th Int. Symp. on Independent Component Analysis and Blind Signal Separation (ICA2003)*, pages 257–262, Nara, Japan, 2003.

[7] H. Valpola, T. Raiko, and J. Karhunen. Building blocks for hierarchical latent variable models. In *Proc. 3rd Int. Conf. on Independent Component Analysis and Signal Separation (ICA2001)*, pages 710–715, San Diego, USA, 2001.

[8] J. Winn and C. M. Bishop. Variational message passing. *J. of Mach. Learn. Res.*, 6:661–694, 2005.